Résolution de contraintes du premier ordre dans des théories dites décomposables

Khalil Djelloul

Laboratoire d'Informatique Fondamentale de Marseille.
Parc scientifique et technologique de Luminy.
163 avenue de Luminy - Case 901.
13288 Marseille, cedex 9. France.
khalil.djelloul@lif.univ-mrs.fr

Résumé Nous présentons dans ce papier un algorithme général pour la résolution de contraintes du premier ordre dans des théories dites *décomposables*. Tout d'abord, en utilisant des quantificateurs spéciaux, nous donnons une caractérisation formelle des théories décomposables et montrons quelques unes de leurs propriétés. Nous présentons en suite un algorithme général pour la résolution de contraintes du premier ordre dans une théorie décomposable T quelconque. L'algorithme est donné sous forme d'un ensemble de cinq règles de réécriture. Il transforme une formule φ, qui bien entendu peut contenir des variables libres, en une conjonction ϕ de formules *résolues*, équivalente dans T et ne faisant pas intervenir d'autre variables libres que celle de φ. Cette conjonction de formules résolues soit est la formule *vrai*, soit est la formule $\neg vrai$, soit a au moins une variable libre. En particulier, si φ n'a pas de variables libres alors ϕ est soit la formule *vrai* soit la formule $\neg vrai$. La correction de notre algorithme démontre la complétude des théories décomposables. Enfin, nous montrons que la théorie $\mathcal{T}$ des arbres finis ou infinis est une théorie décomposable et terminons par une série de benchmarks réalisées par une implantation de notre algorithme, résolvant des formules sur des jeux à deux partenaires dans $\mathcal{T}$ qui font intervenir des imbrications de plus de 160 quantificateurs alternés.

Mots-Clefs. Contraintes logiques du premier ordre, Résolution de contraintes logiques du 1er ordre ; Théorie des arbres ;

1 Introduction

Les arbres éventuellement infinis, jouent un rôle fondamental en informatique. Ils modélisent aussi bien des structures de données que des schémas ou des déroulements de programme. Dès 1976, Gérard Huet a proposé un algorithme pour unifier des termes infinis, c'est-à- dire résoudre des équations dans les arbres [6]. Bruno Courcelle a étudié les propriétés des arbres infinis notamment dans le cadre des schémas récursifs de programme [5,4]. Alain Colmerauer a modélisé le fonctionnement de Prolog II, III et IV par la résolution d'équations et d'inégalités dans les arbres infinis [3,2,1].

Michael J. Maher a axiomatisé tous ces cas par une théorie complète du premier ordre [7]. Il a donc introduit la théorie $\mathcal{T}$ des arbres finis ou infinis ayant un ensemble infini $\mathbf{F}$ de symboles de fonction et a montré sa complétude. C'est cette théorie qui a été le point de départ de nos travaux. Après avoir étudié cette théorie et déduit les propriétés essentielles nous avons créé une nouvelle classe de théories que nous appelons *théories décomposables*. Ces théories présentent des propriétés élégantes et nous ont donc permis de développer un algorithme général efficace pour la résolution de toute contrainte logique du premier ordre dans n'importe quelle théorie décomposable T. Encore faut il se mettre d'accord sur la définition du terme *résolution*. Pour nous, résoudre une contrainte logique φ qui bien entendu peut contenir des variables libres signifie transformer φ en une contrainte logique ϕ, sans introduire de nouvelles variables libres, équivalente a φ dans T, et telle que : (1) soit ϕ est la formule *vrai*, (2) soit ϕ est la formule $\neg$*vrai*, soit ϕ a au moins une variable libre.

Le papier est organisé en cinq sections suivies d'une conclusion. Cette introduction constitue la première section. La deuxième section introduit les notions de base de la logique du premier ordre.

Dans la troisième section, nous présentons notre définition formelle de *théorie décomposable*. L'idée de base de cette définition consiste a doter une théorie T d'un mécanisme élémentaire pour décomposer une suite de quantifications existentielles portant sur une conjonction de formules, en trois suites imbriquées de quantifications ayant des propriétés très particulières, exprimables à l'aide de trois quantificateurs spéciaux notés $\exists?$, $\exists!$, $\exists_\infty^{\Psi(u)}$ et appelés *au-plus-un*, *un-et-un-seul*, *infini*. Ces quantificateurs spéciaux ainsi que leurs propriétés sont décrits en début de section. Alors que les quantificateurs $\exists?$, $\exists!$ ne sont que de simples notations commodes, le quantificateur $\exists_\infty^{\Psi(u)}$, une des clés essentielles de ces théories, exprime une propriété non exprimable au premier ordre.

Dans la section 4, nous présentons notre algorithme de résolution qui résout n'importe quelle formule φ - avec ou sans variables libres - dans n'importe quelle théorie décomposable T. L'algorithme est présenté sous forme d'un ensemble de cinq règles de réécriture. La conjonction ϕ de formules résolues obtenue à partir d'une contrainte φ est équivalente à φ dans T et ne contient pas d'autre variables libres que celles de φ. Nous montrons que ϕ (1) soit est la formule *vrai* donc φ est vrai dans tout modèle de T, (2) soit est la formule $\neg$*vrai* donc φ est fausse dans tout modèle de T, (3) soit contient au moins une variable libre. Nous donnons aussi une approximation de la limite supérieure de la taille de la formule ϕ. Nous montrerons alors que cette taille est bornée par une tour de puissances de 2 évaluée de haut en bas dont la hauteur est la profondeur maximale d'imbrication des négations de la formule φ (si nous utilisons uniquement les symboles $\exists, \wedge$ et $\neg$). La preuve de la correction de notre algorithme montre la complétude de toutes les théories décomposables. Bien entendu, toute théorie décomposable n'admet pas forcement d'élimination de quantificateurs ; citons par exemple la théorie des arbres finis, la théorie des arbres infinis et la théorie des arbres finis ou infinis. Ces trois théories ont été introduites par J. Maher [7] et n'admettent pas

d'élimination de quantificateurs. Cela ne les empêche pas d'être décomposables et donc complètes.

Enfin, nous présentons dans la section 5 la théorie $\mathcal{T}$ des arbres finis ou infinis et montrons qu'elle est décomposable. Nous terminons cette section par des benchmarks réalisés par une implantation de notre algorithme, résolvant des formules sur des jeux à deux partenaires dans $\mathcal{T}$ qui font intervenir des imbrications de plus de 160 quantificateurs alternés. Par manque de place, nous présentons ici uniquement les démonstrations principales, néanmoins, on pourra trouver une version complète de ce papier avec toutes les preuves à l'adresse http ://www.lif.univ-mrs.fr/∼djelloul. Le quantificateur infini, la notion de théorie décomposable, l'algorithme de résolution ainsi que les benchmarks constituent notre contribution essentielle dans ce papier.

2 Rappels sur la logique du premier ordre

2.1 Expressions

On se donne, une bonne fois pour toutes, un ensemble infini dénombrable $\mathbf{V}$ de *variables* et l'ensemble L de symboles *logiques* :

$$=, \mathit{vrai}, \mathit{faux}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists, (,).$$

On se donne aussi, une *signature* S, c'est-à-dire un ensemble de symboles partitionné en deux sous-ensembles : l'ensemble des symboles *de fonction* et l'ensemble des symboles *de relation*. A chaque symbole de fonction et de relation est attaché un entier n positif ou nul, son *arité*. Un symbole n-aire est un symbole d'arité n. Un symbole de fonction 0-aire est appelé *constante*.

Une *expression*, bien entendu, est un mot sur $L \cup S$ qui est soit un *terme*, c'est-à-dire un mot de l'une des deux formes :

$$x,\ ft_1 \dots t_n, \tag{1}$$

soit une *formule*, c'est-à-dire un mot de l'une des onze formes :

$$\begin{array}{c} s = t,\ rt_1 \dots t_n,\ \mathit{vrai},\ \mathit{faux}, \\ \neg\varphi,\ (\varphi \wedge \psi),\ (\varphi \vee \psi),\ (\varphi \rightarrow \psi),\ (\varphi \leftrightarrow \psi), \\ (\forall x\, \varphi),\ (\exists x\, \varphi), \end{array} \tag{2}$$

avec dans (1), x pris dans $\mathbf{V}$, f un symbole de fonction n-aire pris dans S et t_i des termes de tailles plus petites que celui qui est en train d'être défini, et avec dans (2), s, t, t_i des termes, r un symbole de relation n-aire pris dans S et φ et ψ des formules de tailles plus petites que celles qui sont en train d'être définies.

Les formules de la première ligne de (2) sont dites *atomiques*, et *à plat* si elles sont de l'une des formes :

$$\mathit{vrai},\ \mathit{faux},\ x_0 = fx_1 ... x_n,\ rx_1 ... x_n,$$

où les x_i sont des variables prises dans $\mathbf{V}$. On appelle *taille* d'une formule φ et notons $|\varphi|$ le nombre d'occurrences de variables et de symboles de fonction utilisés pour écrire la formule φ.

On rappelle qu'une occurrence d'une variable x dans une formule est *liée* si elle se produit à l'intérieur d'une sous-formule de la forme $(\forall x\, \varphi)$ ou $(\exists x\, \varphi)$. Elle est *libre* dans le cas contraire. Les *variables libres d'une formule* sont celles qui ont au moins une occurrence libre dans cette formule. Une *proposition* est une formule sans variables libres.

Si I est l'ensemble $\{i_1, ..., i_n\}$, on appelle *conjonction* de formules et notons $\bigwedge_{i\in I} \varphi_i$, toute formule de la forme $\varphi_{i_1} \wedge \varphi_{i_2} \wedge ... \wedge \varphi_{i_n} \wedge \mathit{vrai}$. En particulier pour $I = \emptyset$, la conjonction $\bigwedge_{i\in I} \varphi_i$ se réduit à *vrai*. Notons que nous ajoutons la formule *vrai* à la fin de chaque conjonction et ne distinguons pas deux formules qui peuvent être rendues égales moyennant les transformations suivantes des sous-formules :

$$\varphi \wedge \psi \Longrightarrow \psi \wedge \varphi, \;\; (\varphi \wedge \psi) \wedge \phi \Longrightarrow \varphi \wedge (\psi \wedge \phi),$$
$$\varphi \wedge \mathit{vrai} \Longrightarrow \varphi, \;\; \varphi \vee \mathit{faux} \Longrightarrow \varphi.$$

Toutes ces considérations nous seront très utiles dans l'algorithme de résolution présenté à la section 4.

La syntaxe des expressions étant contraignante, on se permet d'utiliser des notations infixées pour les symboles binaires, d'ajouter des parenthèses et d'en enlever quand il n'y a pas d'ambiguïtés.

2.2 Modèle

Un *modèle* est un couple $M = (\mathcal{M}, \mathcal{F})$, où $\mathcal{M}$, *l'univers* ou *domaine* de M, est un ensemble non vide disjoint de S, ses éléments sont appelés *individus* de M ; et $\mathcal{F}$ est une famille d'opérations et de relations dans $\mathcal{M}$.

Si M est un modèle de domaine $\mathcal{M}$, une *M-expression* φ est une expression construite sur la signature $S \cup \mathcal{M}$ au lieu de S en considérant les éléments de $\mathcal{M}$ comme des symboles de fonction d'arité 0. Si pour toute variable libre x de φ, on remplace chaque occurrence libre de x par un même élément de $\mathcal{M}$, on obtient une M-expression appelée *instanciation* de φ par des individus de M.

Si φ est une M-formule, on dit que φ *est vraie dans* M et on écrit

$$M \models \varphi, \tag{3}$$

pour signifier que, pour toute instanciation φ' de φ par des individus de M, l'ensemble $\mathcal{M}$ a la propriété exprimée par $\varphi\prime$, lorsqu'on interprète les symboles de fonction et de relation de $\varphi\prime$ par les fonctions et relations correspondantes de M, et lorsqu'on attribue aux symboles logiques leur sens habituel.

Remarque 2.2.1 *Pour toute M-formule φ sans variables libres, une et une seule des propriétés suivantes est vérifiée : $M \models \varphi$, $M \models \neg\varphi$.*

Terminons cette sous-section par une notation commode. Soit $\bar{x} = x_1...x_n$ un mot sur $\mathbf{V}$ et soit $\bar{i} = i_1...i_n$ un mot sur $\mathcal{M}$ ou $\mathbf{V}$ de la même taille que $\bar{x}$. Si on désigne par $\varphi(\bar{x})$ une M-formule, alors on désignera par $\varphi(\bar{i})$ la M-formule obtenue en remplaçant dans $\varphi(\bar{x})$ chaque occurrence libre de x_j par i_j.

2.3 Théorie

Une *théorie*, est un ensemble (éventuellement infini) de formules sans variables libres, appelées *axiomes*. On dit que le modèle M est *un modèle de* T, si $M \models \varphi$, pour tout élément φ de T. Si φ est une formule, on écrit

$$T \models \varphi,$$

pour signifier que $M \models \varphi$ pour tout modèle M de T. On dit que les formules φ et ψ sont *équivalentes dans* T ssi $T \models \varphi \leftrightarrow \psi$.

2.4 Théorie décomposable

2.5 Quantificateurs vectoriels

Soit M un modèle et T une théorie, tous les deux de signature S. Soient $\bar{x} = x_1 \dots x_n$ et $\bar{y} = y_1 \dots y_n$ deux mots sur $\mathbf{V}$ de même taille et soient ψ, ϕ, φ et $\varphi(\bar{x})$ des M-formules.

Notation 2.5.1 *On écrit*

$$\begin{array}{ll} \exists \bar{x}\, \varphi & \textit{pour } \exists x_1 ... \exists x_n\, \varphi, \\ \forall \bar{x}\, \varphi & \textit{pour } \forall x_1 ... \forall x_n\, \varphi, \\ \exists ? \bar{x}\, \varphi(\bar{x}) & \textit{pour } \forall \bar{x} \forall \bar{y}\, \varphi(\bar{x}) \wedge \varphi(\bar{y}) \rightarrow \bigwedge_{i \in \{1,\dots,n\}} x_i = y_i, \\ \exists ! \bar{x}\, \varphi & \textit{pour } (\exists \bar{x}\, \varphi) \wedge (\exists ? \bar{x}\, \varphi). \end{array}$$

Le mot $\bar{x}$, qui peut être le mot vide ε, est appelé *vecteur de variables*. On remarque que les formules $\exists ? \varepsilon \varphi$ et $\exists ! \varepsilon \varphi$ sont respectivement équivalentes à *vrai* et à φ dans tout modèle.

Propriété 2.5.2 *Si* $T \models \exists ? \bar{x}\, \varphi$ *alors*

$$T \models (\exists \bar{x}\, \varphi \wedge \neg \phi) \leftrightarrow (\exists \bar{x} \varphi) \wedge \neg (\exists \bar{x}\, \varphi \wedge \phi).$$

Propriété 2.5.3 *Si* $T \models \exists ! \bar{x}\, \varphi$ *alors*

$$T \models (\exists \bar{x}\, \varphi \wedge \neg \phi) \leftrightarrow \neg (\exists \bar{x}\, \varphi \wedge \phi).$$

2.6 Quantificateur infini

Soit M un modèle. Soit T une théorie. Soit φ_i et $\varphi(\bar{x})$ deux M-formules et soit $\Psi(u)$ un ensemble de formules ayant au plus u comme variable libre.

Définition 2.6.1 *Nous écrivons*

$$M \models \exists_\infty^{\Psi(u)} x\, \varphi(x), \tag{4}$$

ssi pour toute instanciation $\exists x\, \varphi'(x)$ *de* $\exists x\, \varphi(x)$ *par des individus de* M *et pour tout sous-ensemble* $\{\psi_1(u), .., \psi_n(u)\}$ *d'éléments de* $\Psi(u)$*, l'ensemble des individus* i *de* M *tel que* $M \models \varphi'(i) \wedge \bigwedge_{j \in \{1,\dots,n\}} \neg \psi_j(i)$ *est infini.*
On écrit $T \models \exists_\infty^{\Psi(u)} x\, \varphi(x)$*, ssi pour tout modèle* M *de* T *on a (4).*

Propriété 2.6.2 *Soit J un ensemble fini (éventuellement vide). Si $T \models \exists_\infty^{\Psi(u)} x\, \varphi(x)$ et si pour tout φ_j, une au moins des propriétés suivantes est vérifiée :*
- *$T \models \exists ? x\, \varphi_j$,*
- *il existe $\psi_j(u) \in \Psi(u)$ tel que $T \models \forall x\, \varphi_j \rightarrow \psi_j(x)$,*

alors

$$T \models (\exists x\, \varphi(x) \wedge \bigwedge_{j \in J} \neg \varphi_j)$$

Propriété 2.6.3 *Si $T \models \exists_\infty^{\Psi(u)} x\, \varphi(x)$ alors $T \models \exists_\infty^{\Psi(u)} x$ vrai.*

2.7 Théorie décomposable

Dans tout ce qui suit, nous utiliserons l'abréviation *svls* pour *"sans variables libres supplémentaires"*. Une formule φ est équivalente a une formule svls ψ dans T signifie que $T \models \varphi \leftrightarrow \psi$ et ψ ne contient pas d'autres variables libres que celles de φ.

Définition 2.7.1 *Soit A l'ensemble des conjonctions de formules atomiques à plat. Une théorie T est dite* décomposable *s'il existe un ensemble $\Psi(u)$ de formules ayant au plus u comme variable libre et trois ensembles A', A'' et A''' de formules de la forme $\exists \bar{x} \alpha$ avec $\alpha \in A$ et tels que :*

1. *Toute formule de la forme $\exists \bar{x}\, \alpha \wedge \psi$, avec $\alpha \in A$ et ψ une formule quelconque, est équivalente dans T à une formule svls décomposée de la forme*

$$\exists \bar{x}'\, \alpha' \wedge (\exists \bar{x}''\, \alpha'' \wedge (\exists \bar{x}'''\, \alpha''' \wedge \psi)),$$

   avec $\exists \bar{x}'\, \alpha' \in A'$, $\exists \bar{x}''\, \alpha'' \in A''$ et $\exists \bar{x}'''\, \alpha''' \in A'''$.
2. *Si $\exists \bar{x}' \alpha' \in A'$ alors $T \models \exists ? \bar{x}'\, \alpha'$ et pour toute variable libre y dans $\exists \bar{x}' \alpha'$, une au moins des propriétés suivantes est satisfaite :*
   - *$T \models \exists ? y \bar{x}'\, \alpha'$,*
   - *il existe $\psi(u) \in \Psi(u)$ tel que $T \models \forall y\, (\exists \bar{x}'\, \alpha') \rightarrow \psi(y)$.*
3. *Si $\exists \bar{x}'' \alpha'' \in A''$ alors pour tout x_i'' de $\bar{x}''$ on a $T \models \exists_\infty^{\Psi(u)} x_i''\, \alpha''$.*
4. *Si $\exists \bar{x}''' \alpha''' \in A'''$ alors $T \models \exists ! \bar{x}'''\, \alpha'''$.*
5. *Si la formule $\exists \bar{x}' \alpha'$ appartient a A' et n'a pas de variables libres alors cette formule est soit la formule $\exists \varepsilon$vrai soit la formule $\exists \varepsilon$faux.*

Propriété 2.7.2 *Soit T une théorie décomposable. Toute formule de la forme $\exists \bar{x}\, \alpha$, avec $\alpha \in A$, est équivalente dans T à une formule svls de la forme $\exists \bar{x}'\, \alpha'$ avec $\exists \bar{x}' \alpha' \in A'$.*

En utilisant le quatrième point de la définition 2.7.1 nous obtenons la propriété suivante

Corollaire 2.7.3 *Soit T une théorie décomposable. Toute formule sans variables libres de la forme $\exists \bar{x}\, \alpha$, avec $\alpha \in A$, est équivalente dans T soit à vrai soit à faux.*

3 Un algorithme général pour la résolution de contraintes du premier ordre dans une théorie décomposable T

Soit T une théorie décomposable. Les ensembles $\Psi(u)$, A', A'' et A''' sont maintenant connus et fixés. Notons A l'ensemble des conjonctions de formules atomiques à plat dans T.

3.1 Formules normalisées

Définition 3.1.1 *Une formule normalisée φ de profondeur $d \geq 1$ est une formule de la forme*

$$\neg(\exists \bar{x}\, \alpha \wedge \bigwedge_{i \in I} \varphi_i), \tag{5}$$

avec I un ensemble fini (éventuellement vide), $\alpha \in A$ et φ_i des formules normalisées de profondeur d_i avec $d = 1 + \max\{0, d_1, ..., d_n\}$.

Il est facile de transformer n'importe quelle formule du premier ordre en une formule normalisée équivalente dans T. Pour cela il suffit par exemple : (1) d'introduire un supplément d'équations et de variables quantifiées pour transformer toute conjonction d'équations en une conjonction d'équations atomiques à plat, (2) d'exprimer tous les quantificateurs, constantes et connecteurs logique avec $\neg$, $\wedge$ et $\exists$, (3) de supprimer les doubles négations : $\neg\neg\varphi$ devient φ, (4) si la formule φ obtenue ne commence pas $\neg$, il suffit de la remplacer par $\neg\neg\varphi$, (5) de nommer les variables quantifiées par des noms différents de ceux des variables libres, et aussi différents que possible, (6) de remonter la quantification existentielle au dessus des conjonctions : $\varphi \wedge (\exists \bar{x}\, \psi)$ devient $(\exists \bar{x}\, \varphi \wedge \psi)$ car les variables libres de φ sont distincts de ceux de $\bar{x}$. Si la formule de départ ne contient par le symbole logique $\leftrightarrow$ alors cette transformation est linéaire c'est-à-dire qu'il existe une constante k telle que $n_2 \leq k n_1$, avec n_1 la taille de la formule de départ et n_2 la taille de la formule normalisée finale.

Définition 3.1.2 *Une formule* résolue *est une formule normalisée de la forme*

$$\neg(\exists \bar{x}'\, \alpha' \wedge \bigwedge_{i \in I} \neg(\exists \bar{y}'_i\, \beta'_i)), \tag{6}$$

où I est un ensemble fini (éventuellement vide), $\exists \bar{x}' \alpha' \in A'$, $\exists \bar{y}'_i \beta'_i \in A'$, α' différent de la formule faux et tous les β'_i sont différents des formules vrai et faux.

En utilisant le dernier point de la définition 2.7.1 nous montrons la propriété suivante :

Propriété 3.1.3 *Soit φ une conjonction de formules résolues sans variables libres. La conjonction φ est soit la formule $\neg(\exists \varepsilon$ vrai$)$ soit la formule vrai.*

3.2 Les règles de réécriture

Nous présentons maintenant les règles de réécriture qui transforment une formule normalisée de profondeur quelconque en une conjonction de formules résolues, équivalente dans T. Appliquer la règle $p_1 \Longrightarrow p_2$ à la formule normalisée p signifie remplacer dans p, une sous-formule p_1 par la formule p_2, en considérons le connecteur $\wedge$ associatif et commutatif.

$$(1) \quad \neg \begin{bmatrix} \exists \bar{x}\, \alpha \wedge \varphi \wedge \\ \neg(\exists \bar{y}\, vrai) \end{bmatrix} \Longrightarrow vrai$$

$$(2) \quad \neg \begin{bmatrix} \exists \bar{x}\, faux \wedge \varphi \end{bmatrix} \Longrightarrow vrai$$

$$(3) \quad \neg \begin{bmatrix} \exists \bar{x}\, \alpha \wedge \\ \bigwedge_{i \in I} \neg(\exists \bar{y}_i\, \beta_i) \end{bmatrix} \Longrightarrow \neg \begin{bmatrix} \exists \bar{x}' \bar{x}''\, \alpha' \wedge \alpha'' \wedge \\ \bigwedge_{i \in I} \neg(\exists \bar{x}''' \bar{y}_i\, \alpha''' \wedge \beta_i) \end{bmatrix}$$

$$(4) \quad \neg \begin{bmatrix} \exists \bar{x}\, \alpha \wedge \\ \bigwedge_{i \in I} \neg(\exists \bar{y}'_i\, \beta'_i) \end{bmatrix} \Longrightarrow \neg \begin{bmatrix} \exists \bar{x}'\, \alpha' \wedge \\ \bigwedge_{i \in I'} \neg(\exists \bar{y}'_i\, \beta'_i) \end{bmatrix}$$

$$(5) \quad \neg \begin{bmatrix} \exists \bar{x}\, \alpha \wedge \varphi \wedge \\ \neg \begin{bmatrix} \exists \bar{y}'\, \beta' \wedge \\ \bigwedge_{i \in I} \neg(\exists \bar{z}'_i\, \delta'_i) \end{bmatrix} \end{bmatrix} \Longrightarrow \begin{bmatrix} \neg(\exists \bar{x}\, \alpha \wedge \varphi \wedge \neg(\exists \bar{y}'\, \beta')) \wedge \\ \bigwedge_{i \in I} \neg(\exists \bar{x} \bar{y}' \bar{z}'_i\, \alpha \wedge \beta' \wedge \delta'_i \wedge \varphi) \end{bmatrix}$$

avec $\alpha \in A$, φ une conjonction de formules normalisées et I un ensemble fini (éventuellement vide). Dans la règle (3), la formule $\exists \bar{x}\, \alpha$ est équivalente dans T à une formule décomposée de la forme $\exists \bar{x}'\, \alpha' \wedge (\exists \bar{x}''\, \alpha'' \wedge (\exists \bar{x}'''\, \alpha'''))$ avec $\exists \bar{x}'\, \alpha' \in A'$, $\exists \bar{x}''\, \alpha'' \in A''$, $\exists \bar{x}'''\, \alpha''' \in A'''$ et $\exists \bar{x}'''\, \alpha'''$ différent de la formule $\exists \varepsilon\, vrai$. Tous les β_i sont des éléments de A. Dans la règle (4), la formule $\exists \bar{x}\, \alpha$ n'est pas un élément de A' et est équivalente dans T à formule décomposée de la forme $\exists \bar{x}'\, \alpha' \wedge (\exists \bar{x}''\, \alpha'' \wedge (\exists \varepsilon\, vrai))$ avec $\exists \bar{x}'\, \alpha' \in A'$ et $\exists \bar{x}''\, \alpha'' \in A''$. Chaque formule $\exists \bar{y}'_i\, \beta'_i$ est un élément de A'. I' est l'ensemble des $i \in I$ tels que $\exists \bar{y}'_i \beta'_i$ ne contient aucune occurrences libres de n'importe quelle variable de $\bar{x}''$. Dans la règle (5), $I \neq \emptyset$, $\exists \bar{y}'\, \beta' \in A'$ et $\exists \bar{z}'_i\, \delta'_i \in A'$.

Propriété 3.2.1 *Toute application répétée de ces règles de réécriture sur une formule normalisée φ, se termine et produit une conjonction svls ϕ de formules résolues équivalente à φ dans T. De plus, $|\phi|$ est bornée supérieurement par une tour de puissances de 2 évaluée de haut en bas dont la hauteur est la profondeur maximale d'imbrication des négations de la formule φ*

3.3 L'algorithme de résolution

Soit ψ une formule quelconque. La résolution de ψ consiste à :

1. Transformer la formule ψ en une formule normalisée φ équivalente à ψ dans T.

2. Appliquer les règles de réécriture précédentes sur φ autant de fois que possible. A la fin nous obtenons une conjonction ϕ de formules résolues.

D'après la propriété 3.2.1, l'application des règles de réécriture sur une formule ψ sans variables libres produit une conjonction équivalente svls ϕ de formules résolues. Du fait que ϕ est svls et que ψ ne contient pas de variables libres alors ϕ ne contient pas de variables libres. D'après la propriété 3.1.3, ϕ est soit la formule *vrai*, soit la formule $\neg(\exists\varepsilon vrai)$, donc soit $T \models \neg\psi$, soit $T \models \neg\psi$ et donc T est une théorie complète.

Nous pouvons maintenant énoncer notre résultat principal :

Théorème 3.3.1 *Si T est une théorie décomposable alors toute formule est équivalente dans T soit à vrai, soit à faux soit à une conjonction svls de formules résolues ayant au moins une variable libre.*

Ce théorème montre l'importance de la notion de *théorie décomposable* et établit la connection entre *théorie décomposable* et *théorie complète*. En effet, nous disposons maintenant d'une nouvelle condition suffisante de complètude :

Théorème 3.3.2 *Si T est une théorie décomposable alors T est une théorie complète.*

4 Résolution de contraintes du premier ordre dans la théorie $\mathcal{T}$ des arbres finis ou infinis

4.1 Les axiomes

La théorie $\mathcal{T}$ des arbres finis ou infinis a pour ensemble $\mathbf{F}$ de symboles de fonction, un ensemble infini comprenant au moins un symbole d'arité non nulle et pour ensemble $\mathbf{R}$ de symboles de relation, l'ensemble vide. Elle consiste en l'ensemble infini de propositions de l'une des trois formes suivantes :

$$\begin{array}{lll} \forall\bar{x}\forall\bar{y} & \neg f\bar{x} = g\bar{y} & [1] \\ \forall\bar{x}\forall\bar{y} & f\bar{x} = g\bar{y} \rightarrow \bigwedge_i x_i = y_i & [2] \\ \forall\bar{x}\exists!\bar{y} & \bigwedge_i y_i = t_i[\bar{x}\bar{y}] & [3] \end{array}$$

où f et g sont deux symboles de fonction distincts pris dans $\mathbf{F}$, $\bar{x}$ est un vecteur composé variables x_i, $\bar{y}$ est un vecteur de variables y_i toutes distinctes et $t_i[\bar{x}\bar{y}]$ est un terme qui commence par un élément de $\mathbf{F}$ suivis de variables prises dans $\bar{x}$ ou $\bar{y}$.

4.2 Propriétés des conjonctions d'équations à plat dans $\mathcal{T}$

Supposons que l'ensemble $\mathbf{V}$ des variables est totalement ordonné par une relation d'ordre total et dense notée $\succ$.

Définition 4.2.1 *Une conjonction α d'équations à plat est dite* ($\succ$)-résolue *si tous ces membres gauches sont distincts et α ne contient pas d'équations de la forme $x = x$ ou $x = y$, avec x et y des variables telles que $y \succ x$.*

Propriété 4.2.2 *Toute conjonction α de formule à plat est équivalente dans $\mathcal{T}$ soit à faux soit à une conjonction $(\succ)$-résolue d'équations à plat.*

Introduisons maintenant la notion de variables et équations *accessibles*.

Définition 4.2.3 *Les équations et variables accessibles à partir de la variable u dans la formule $\exists\bar{x}\bigwedge_{i=1}^{n} v_i = t_i$ sont celles qui apparaissent dans au moins une des ses sous-formules de la forme $\bigwedge_{j=1}^{m} v_{k_j} = t_{k_j}$, où v_{k_1} est la variable u et v_{k_j+1} apparaît dans le terme t_{k_j} pour tout $j \in \{1,..,m\}$. Les équations et variables accessibles de cette formule sont celles qui sont accessibles à partir d'une variable qui n'apparaît pas dans $\bar{x}$.*

Example 1. Dans la formule

$$\exists uvw\, z = fuv \wedge v = gvu \wedge w = fuv,$$

l'équation $z = fuv$ et $v = gvu$ et les variables u et v sont accessibles. Par contre, l'équation $w = fuv$ et la variable w ne le sont pas.

D'après les axiomes [1] et [2] de $\mathcal{T}$ nous avons la propriété suivante :

Propriété 4.2.4 *Soit α une conjonction d'équations à plat. Si toutes les variables de $\bar{x}$ sont accessibles dans $\exists\bar{x}\,\alpha$ alors $\mathcal{T} \models \exists?\bar{x}\,\alpha$.*

D'autre part, en utilisant l'axiome [3] nous obtenons :

Propriété 4.2.5 *Soit α une conjonction $(\succ)$-résolue d'équations à plat et soit $\bar{x}$ le vecteur de ses membres gauches. Nous avons $\mathcal{T} \models \exists!\bar{x}\,\alpha$.*

4.3 Résolution de contraintes du premier ordre dans $\mathcal{T}$

Propriété 4.3.1 *$\mathcal{T}$ est une théorie décomposable .*

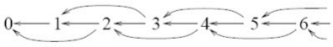
Pour montrer cette propriété nous choisissons nos ensembles $\Psi(u)$, A', A'' et A''' de la manière suivante :

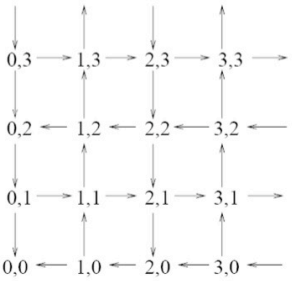
- $\Psi(u)$ est l'ensemble des formules de la forme $\exists\bar{y}\, u = f\bar{y}$ avec $f \in \mathbf{F}$,
- A' est l'ensemble des formules de la forme $\exists\bar{x}'\alpha'$ telles que
  - α' est soit la formule *faux*, soit une conjonction $(\succ)$-résolue d'équations à plat et où l'ordre $\succ$ est tel que toutes les variables de $\bar{x}'$ sont supérieures aux variables libres de $\exists\bar{x}'\alpha'$,
  - toutes les variables de $\bar{x}'$ et toutes les équations de α' sont accessibles dans $\exists\bar{x}'\alpha'$,
- A'' est l'ensemble des formules de la forme $\exists\bar{x}''$ *vrai*,
- A''' est l'ensemble des formules de la forme $\exists\bar{x}'''\alpha'''$ telles que α''' est une conjonction $(\succ)$-résolue d'équations à plat et $\bar{x}'''$ est le vecteur des membres gauches des équations de α'''.

Nous montrons par induction sur la structure des conjonctions $(\succ)$-résolues d'équations à plat que $\mathcal{T}$ satisfait les cinq conditions de la définition 2.7.1.

Benchmarks : Jeux à deux partenaires

Soit (V, E) un graphe orienté, avec V l'ensemble des sommets et $E \subseteq V \times V$ un ensemble d'arêtes. Les ensembles V et E peuvent être vides et les éléments de E sont aussi appelés *positions*. Considérons un jeu à deux partenaires qui étant donnée une position initiale x_0, consiste à tour de rôle à choisir une position x_1 telle que $(x_0, x_1) \in E$, puis une position x_2 telle que $(x_1, x_2) \in E$ etc... Le premier qui ne peut plus jouer a perdu et l'autre a gagné. Par exemple les deux graphes suivants correspondent aux deux jeux suivants :





Jeux 1 On se donne un entier positif ou nul i et à tour de rôle, on soustrait 1 ou 2 de i sans jamais le rendre strictement négatif. La première personne qui ne peut plus jouer a perdu.

Jeux 2 On se donne un couple (i, j) d'entiers positifs ou nuls et à tour de rôle, on choisit l'un des deux entiers i, j. Suivant que l'entier choisi u est impair ou pair, on augmente ou on diminue l'autre entier v de 1 sans jamais le rendre strictement négatif. La première personne qui ne peut plus jouer a perdu.

Soit x une position dans un jeux et supposons que c'est au tour de la personne A de jouer. La position x est dite k-gagnante si, quelque soit la façon de jouer de l'autre personne B, il est toujours possible que A gagne en jouant au plus k coups. La position x est dite k-perdante si quelque soit la façon de jouer de A, il existe toujours une façon de jouer pour B qui a pour effet que A perde et joue au plus k coups. Nous montrons sans trop de difficultés que

$$gagnant_k(x) = \begin{bmatrix} \exists y\, coup(x,y) \wedge \neg(\\ \exists x\, coup(y,x) \wedge \neg(\\ ... \\ \exists y\, coup(x,y) \wedge \neg(\\ \exists x\, coup(y,x) \wedge \neg(\\ faux \qquad \underbrace{)...)}_{2k} \end{bmatrix}$$

où coup(x, y) signifie : " à partir d'une position x on joue une fois et on atteint la position y". En descendant les négations, on obtient ainsi une imbrication de $2k$ quantificateurs alternés. Nous présentons ces deux jeux dans la théorie des arbres finis ou infinis $(A, \mathbf{F})$, où chaque position est représentée par un arbre. Si nous prenons comme entrée de notre solver la formule $gagnant_k(x)$, nous obtenons en sortie une formule qui représente toutes les positions k-gagnantes.

Jeux 1 : Supposons que $\mathbf{F}$ contienne le symbole de fonction 0 d'arité 0 et le symbole de fonction s d'arité 1. Nous codons les sommets i du graphe de ce jeu par l'arbre $s^i(0)$.[1] La relation $coup(x, y)$ est définie comme suit :

$$coup(x, y) \overset{\text{def}}{\leftrightarrow} x = s(y) \vee x = s(s(y)) \vee (\neg(x = 0) \wedge \neg(\exists u\, x = s(u)) \wedge x = y)$$

Pour $gagnant_1(x)$ notre algorithme donne la formule résolue suivante :

$$\neg \left[\exists \varepsilon\ vrai \wedge \begin{bmatrix} \neg(\exists u\, x = s(u) \wedge u = 0) \wedge \\ \neg(\exists u_1 u_2\, x = s(u_1) \wedge u_1 = s(u_2) \wedge u_2 = 0) \end{bmatrix} \right]$$

qui correspond à la solution $x = s(0) \vee x = s(s(0))$.
Jeux 2 : Supposons que $\mathbf{F}$ contienne les symboles de fonction 0, f, g, c d'arité respective 0, 1, 1, 2. Nous codons les sommets (i, j) du graphe de ce jeu par l'arbre $c(\overline{i}, \overline{j})$ avec $\overline{i} = (fg)^{i/2}(0)$ si i est paire et $\overline{i} = g(\overline{i-1})$ si i est impaire.[2] La relation $coup(x, y)$ est définie comme suit :

$$coup(x, y) \overset{\text{def}}{\leftrightarrow} transition(x, y) \vee (\neg(\exists uv\, x = c(u, v)) \wedge x = y)$$

avec

$$transition(x, y) \overset{\text{def}}{\leftrightarrow} \begin{bmatrix} \exists uvw \\ \begin{bmatrix} (x = c(u, v) \wedge y = c(u, w)) \vee \\ (x = c(v, u) \wedge y = c(w, u)) \end{bmatrix} \\ \wedge \\ \begin{bmatrix} (\exists i\, u = g(i) \wedge succ(v, w)) \vee \\ (\neg(\exists i\, u = g(i)) \wedge pred(v, w)) \end{bmatrix} \end{bmatrix}$$

$$succ(v, w) \overset{\text{def}}{\leftrightarrow} \begin{bmatrix} ((\exists j\, v = g(j) \wedge w = f(v)) \vee \\ (\neg(\exists j\, v = g(j)) \wedge w = g(v)) \end{bmatrix}$$

$$pred(v, w) \overset{\text{def}}{\leftrightarrow} \begin{bmatrix} (\exists j\, v = f(j) \wedge \begin{bmatrix} (\exists k\, j = g(k) \wedge w = j) \vee \\ (\neg(\exists k\, j = g(k)) \wedge w = v) \end{bmatrix}) \vee \\ (\exists j\, v = g(j) \wedge \begin{bmatrix} (\exists k\, j = g(k) \wedge w = v) \vee \\ (\neg(\exists k\, j = g(k)) \wedge w = j) \end{bmatrix}) \vee \\ (\neg(\exists j\, v = f(j)) \wedge \neg(\exists j\, v = g(j)) \wedge \neg(v = 0) \wedge w = v) \end{bmatrix}$$

Pour $gagnant_1(x)$ notre algorithme donne la formule résolue suivante :

$$\neg \left[\exists \varepsilon\ vrai \wedge \begin{bmatrix} \neg(\exists u_1 u_2 u_3\, x = c(u_1, u_2) \wedge u_1 = g(u_3) \wedge u_2 = 0 \wedge u_3 = 0) \wedge \\ \neg(\exists u_1 u_2 u_3\, x = c(u_1, u_2) \wedge u_2 = g(u_3) \wedge u_1 = 0 \wedge u_3 = 0) \end{bmatrix} \right]$$

qui correspond à la solution $x = c(g(0), 0) \vee x = c(0, g(0))$.

Les temps d'exécution (CPU time en millisecondes) des formules $gagnant_k(x)$ sont résumés dans le tableau suivant. L'algorithme a été programmé en C++ et

[1] Bien entendu $s^0(x) = x$ et $s^{i+1} = s(s^i(x))$.
[2] $(fg)^0(x) = x$ et $(fg)^{i+1}(x) = f(g((fg)^i(x)))$.

les benchmarks sont réalisés sur un processeur 2.5Ghz Pentium IV , avec 1024Mb de RAM.

k	0	1	2	4	10	20	40	80
jeux 1	0	0	5	10	150	2130	45430	1920110
jeux 2	0	75	180	420	3040	123025	–	–

Notons que nous sommes capable de calculer les k-gagnantes positions du jeux 1 avec k = 80, ce qui correspond à résoudre une formule normalisée avec 160 quantificateurs alternés et imbriqués.

5 Conclusion

Nous avons défini dans ce papier une nouvelle classe de théories que nous appelons *théories décomposables* et avons donné un algorithme général pour la résolution de toute contrainte du premier ordre dans ces théories. Cet algorithme est donné sous forme d'un ensemble de cinq règles de réécriture et sa correction constitue la preuve de la complétude des théories décomposables.

Nous avons aussi montré que la théorie $\mathcal{T}$ des arbres finis ou infinis est une théorie décomposable. S. Vorobyov [8] a montré que le problème de décision si une proposition sans variables libres est vraie ou non dans $\mathcal{T}$ est non élémentaire, c'est-à-dire que la complexité de tout algorithme qui le résout n'est pas borné par une tour de puissances de 2 avec une hauteur fixe. Donc, il est tout à fait normal que la complexité de notre algorithme et la taille de nos formules résolues sont de cet ordre. Cependant, malgré cette haute complexité, nous avons implanté notre algorithme et résolu des benchmarks dans $\mathcal{T}$ faisant intervenir plus de 160 quantificateurs alternés et imbriqués.

Actuellement, nous établissons une longue liste de théories décomposables telles que la combinaison des arbres et des rationnels additifs. Nous travaillons également sur une axiomatisation complète dans une théorie décomposable plus élaborés d'un mélange de la théorie des arbres finis ou infinis avec la théorie des réels munis des opérations d'addition " $+$ ", soustraction, " $-$ ", multiplication " $*$ " et d'une relation d'ordre dense sans extrêmes. Nous essayons aussi de développer quelques méthodes intuitives pour trouver facilement les ensembles $\psi(u)$, A', A'' et A'''.

Remerciements Je remercie Alain Colmerauer pour nos nombreuses discussions et son aide dans l'organisation et la rédaction de ce papier. Je le remercie aussi pour les définitions et démonstrations de ses notes de cours de DEA. Je lui dédie ce papier en lui souhaitant un rétablissement rapide.

Références

1. Benhamou F, Colmerauer A, Garetta H, Pasero R, Van Caneghem M. Le manuel de Prolog IV , PrologIA, Marseille, France, 1996.
2. Colmerauer A. An introduction to Prolog III. *Communication of the ACM*, 33(7) :68–90,1990.

3. Colmerauer A. Equations et inequations on finite et infinite trees. Proceeding of the International conference on the fifth generation of computer systems Tokyo, 1984. P. 85–99.
4. Courcelle B. Equivalences et Transformations of Regular Systems applications to Program Schemes et Grammars, Theretical Computer Science, vol. 42, 1986, p. 1–122.
5. Courcelle B. Fundamental Properties of Infinite Trees, Theoretical Computer Science, vol. 25, n o 2, 1983, p. 95–169.
6. Huet G. Resolution d'equations dans les langages d'ordre 1, 2,...ω. These d'Etat, Universite Paris 7. France,1976.
7. Maher M. Complete axiomatization of the algebra of finite, rational et infinite trees. *Technical report, IBM - T.J.Watson Research Center*, 1988.
8. Vorobyov S. An Improved Lower Bound for the Elementary Theories of Trees, Proceeding of the 13th International Conference on Automated Deduction (CADE'96). Springer Lecture Notes in Artificial Intelligence, vol 1104, pp. 275- 287, New Brunswick, NJ, July/August, 1996.